What is this Fuego thing and Where is it going?

Tim Bird

Fuego Test System Maintainer

Sr. Staff Software Engineer, Sony Electronics

Outline

Vision
Major elements
Architecture
Details
Roadmap

Outline

Vision

Major elements

Architecture

Details

Roadmap

Vision – super high level

Do for testing
what open source
has done for coding

- Fuego Goal:
  - *Promote the sharing of tests, test methods, and results, the way code is shared now*
    - Make it easy to create, share and discover tests
    - Make test results easy to share and evaluate

Fuego is a test framework focused on tests and test re-use

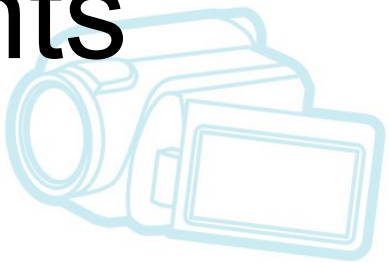
Outline

Vision

Major elements

Architecture

Details



Major Elements

- Jenkins interface
- Test execution system
- Pre-packaged tests
- Docker container
- Test distribution

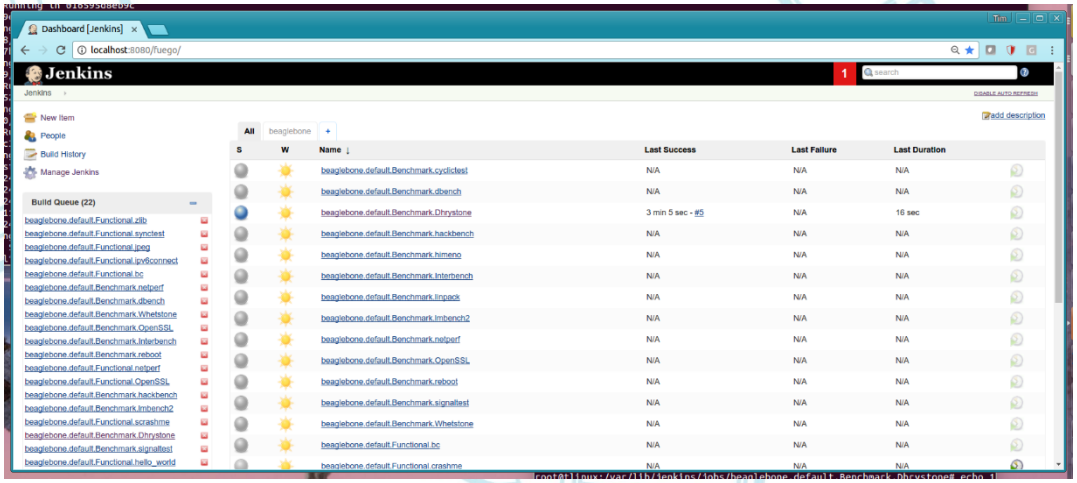
Jenkins

- Is a Continuous Integration system
- Handles job scheduling, notifications, visualization, etc.
  - Launches test jobs based on various triggers
  - Shows test results
  - Has an very large ecosystem of plugins for lots of extended functionality
    - E-mail notifications
    - Plotting of results
    - Integration with different source code management systems
- Is too big a system to describe in detail here

Jenkins

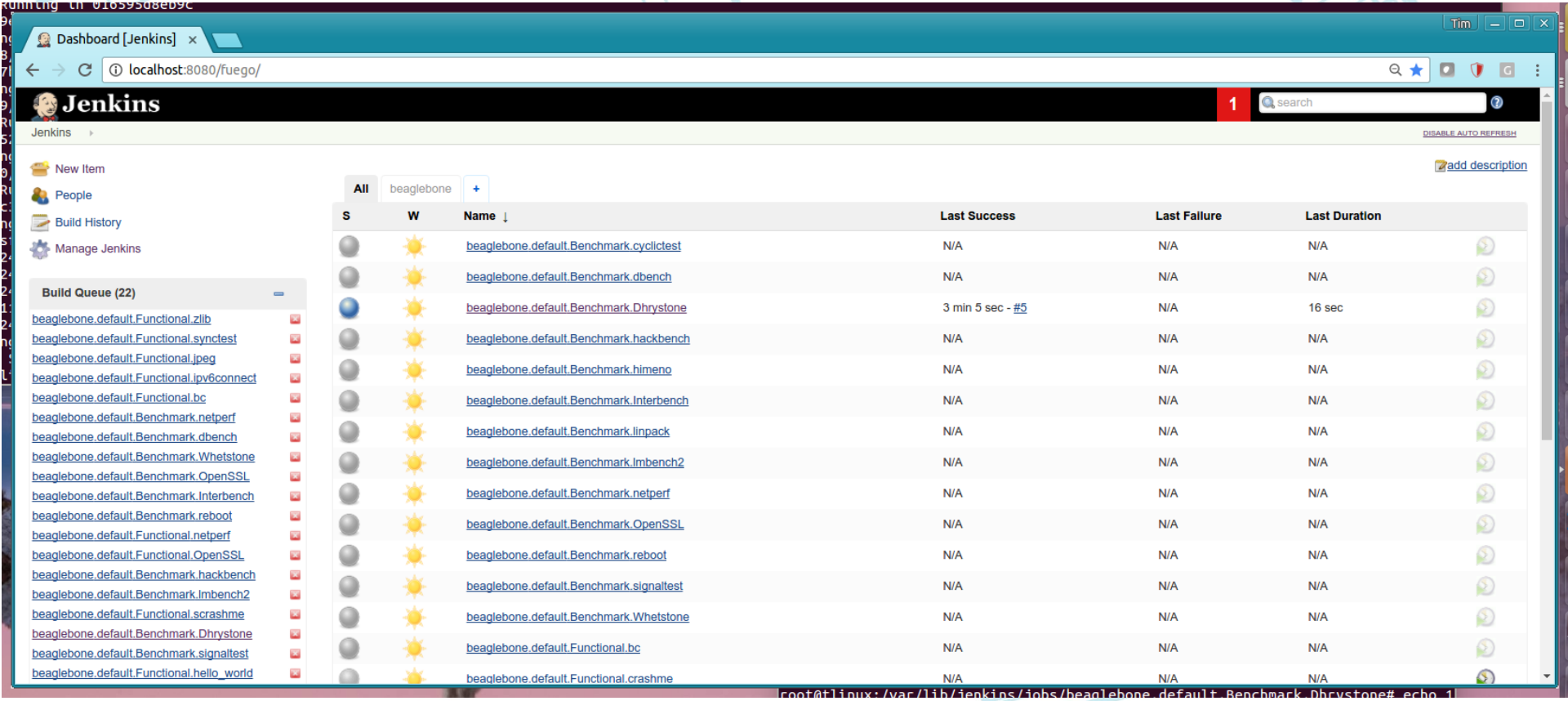
- Base interface:

  Build queue and job list



- Fuego provides instances of Jenkins objects
  - Jenkins node = Fuego board (Device under test)
  - Jenkins job = Fuego test
- Fuego has a custom plugin for plotting

A closer look



Test execution system

- Runs the test
  - Constructs the test environment
  - Executes the test in multiple phases
    - Build, deploy, run, process results
  - Evaluates results
- Fuego system allows customization
  - Test parameters
  - Results evaluation criteria
  - Board and platform management functions
- Test results are converted into a unified format

Pre-packaged tests

- Comes with over 100 tests, already integrated
  - aim7, aiostress, arch_timer, blobsalad, bonnie, bzip2, cmt, crashme, cyclictest, dbench, dhrystone, ebizzy, , expat, ffsb, fio, fontconfig, glib, GLMark, gtkperf, hackbench, himeno, Interbench, IOzone, iperf, ipv6connect, Java, jpeg, kernel_build, kselftest, libpng, linpack, linus_stress, lmbench2, LTP, nbench, netperf, netpipe, OpenSSL, posixtestsuite, reboot, rmaptest, signaltest, Stream, scifab, scrashme, sdhi_o, stress, synctest, tiobench, whetstone, x11perf, zlib, and many others
  - Goal is to have thousands of tests for user to choose from
- Test categories:
  - File system, networking, real-time, graphics, cpu performance, Posix conformance, etc.
- Includes functional and benchmark tests

Inside a container

- Fuego builds a docker container during installation
  - Container has auxiliary programs and libraries needed for test system
- This avoids install issues
  - Fuego can run on any Linux distro
- Used as a controlled execution environment
  - Builds of the test programs are reproducible
- Provides an isolated environment for security
  - Portions of tests execute on the host machine
  - Container prevents a test from disrupting the host

Fuego is a Linux distribution

- Distribution inside the container has all the tools that a host needs to conduct tests
  - Toolchains
  - Servers
  - Libraries
  - Emulators
  - Parsers
  - Test software itself
- Goal is to curate testing resources for the user (QA engineer)

Outline

Vision

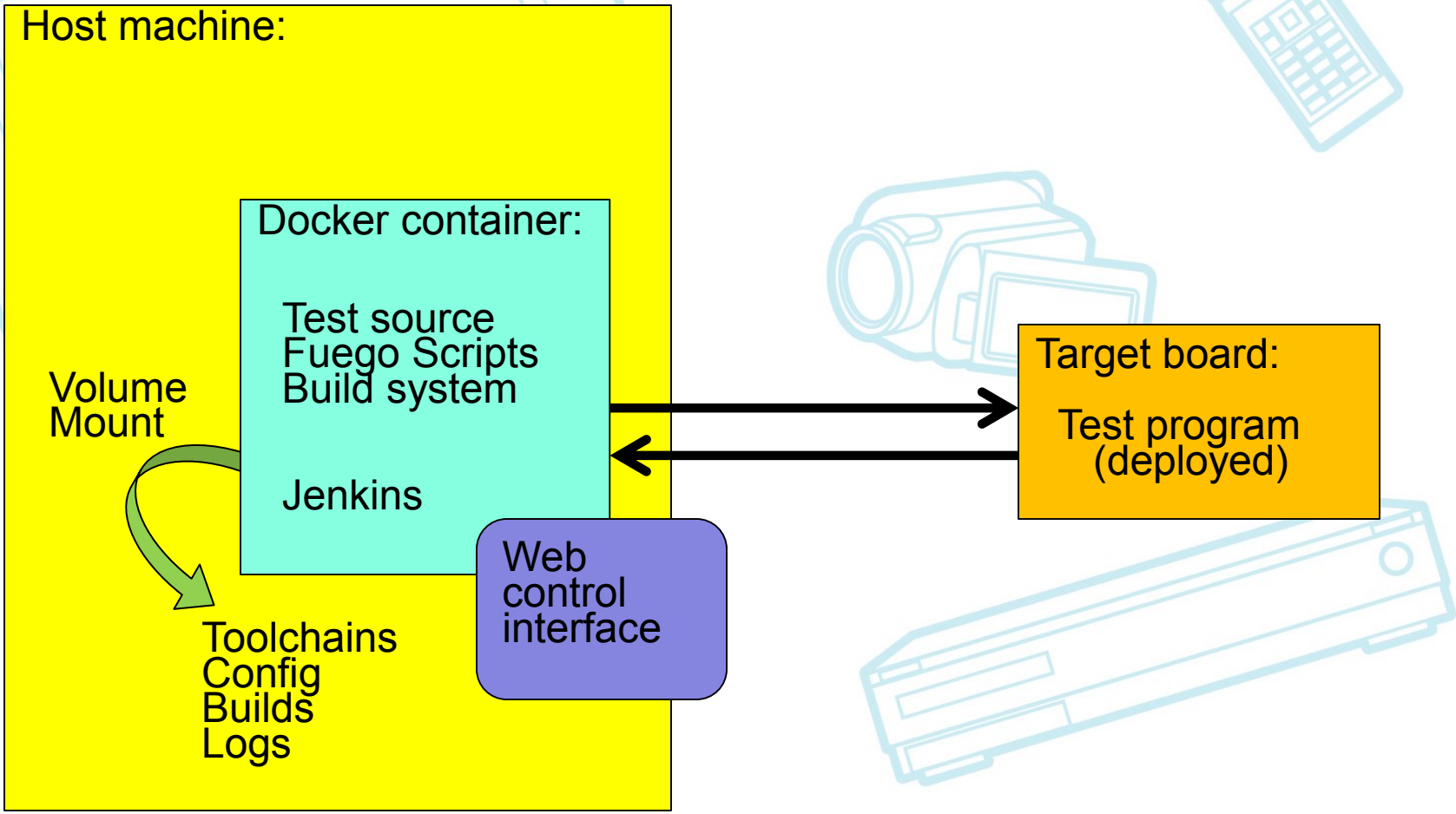
Major elements

Architecture

Details

Roadmap

Architecture Diagram



Architecture

- Is intrinsically host/target
  - Software is cross-compiled
  - Test execution is directed by a host machine
  - Minimal requirements for the Device Under Test
    - e.g. Posix shell, a few common utilities (e.g. cat, grep)
    - No assumptions about interpreters, libraries, etc:
      - Only run shell script or natively compiled code
- Fuego can wrap existing test programs
  - E.g. Dhyrstone, hackbench, LTP
- Or execute new custom programs

Division of roles

- Layering of functionality:
  - Docker provides overall build and execution environment
  - Jenkins handles job scheduling and results presentation
  - Fuego scripts build the test program, deploy it to the target, and post-process results
  - Test program executes on the target board
- User can do operations at command line

Outline

Vision

Major elements

Architecture

Details

Roadmap

Fuego details

- Test building
- Fuego source
- Installation
- Fuego abstractions and customization
- Using Fuego (outline)
- Test elements

Test building

- Tests are built from source
  - You provide your toolchain (SDK)
    - Can use a pre-installed generic toolchain in some cases
- Fuego supports any target architecture
- If test program is a simple shell program, a build is not needed
- Resulting program is transferred to target during test
  - No need to have it pre-installed
    - But it can be

Fuego source

- Comes as 2 git repositories:
  - 'fuego' repository - Stuff outside the container
    - Container build system
      - Including some Jenkins plugins
    - Default config and boards
    - Host scripts for controlling the container
    - Documentation
  - 'fuego-core' repository - Stuff inside the container
    - Script and overlay engine
    - Pre-packaged tests
    - More jenkins extensions

Installation

- Simple installation
  - Requirements met by packages installed in container
- Steps:

```
$ git clone https://bitbucket.org/fuegotest/fuego.git
$ git clone https://bitbucket.org/fuegotest/fuego-core.git
$ cd fuego ; ./install.sh
        (wait a bit while docker image and container are created)
$ ./start.sh
$ ftc add-node –b localhost
        (then, populate the interface with the jobs you want to run)
$ ftc add-jobs –b localhost –p testplan_default
      $ chrome http://localhost:8080/fuego
```

Fuego Abstractions

- Fuego tests and boards are simple shell scripts:
  - Board: Variables describing board properties
  - Tests: Functions to build, deploy, execute test
- Fuego provides functions for command and control of target:
  - Put/get files, execute commands, collect logs, etc.
- Fuego customizes test variables at runtime
  - Based on board information, toolchain configuration, and test variant definitions
  - This allows any aspect of a test to be abstracted

Test parameter abstraction

- Test variables (or parameters) allows a test to be run in different variations.
- Fuego abstracts target access methods
  - get, put files, command execution
- Fuego also abstracts:
  - Toolchain for software builds
  - Filesystem block device names and mount points
- User can add new items to be abstracted
- Test plan system allows a single test to be run in multiple variations
  - A test variant is called a ‘spec’

Using Fuego (outline)

- To use existing tests:
  - Add a board configuration
  - Add a toolchain
  - Select tests to run
    - Add jobs for tests to the Jenkins interface
  - Configure job triggers
- To customize existing tests:
  - Adjust test parameters
  - Customize results processing

Test elements

- fuego_test.sh
  - Simple shell file with functions called by fuego:
    - test_build
    - test_deploy
    - test_run
  - Also indicates test program source
    - Usually a tarball, but can be a git repo reference
- May include a python program for parsing results from the test log file
  - Results are converted to structured data
- May include a file to describe pass criteria
- May include a file with test metadata

Outline

Vision
Major elements
Architecture
Details
Roadmap

Where is it going?

Staying in my lane...

- I don't want to add to Fuego:
  - Email-based patch CI triggers
  - SUT image provisioning
  - Board control drivers/lab management software
  - Centralized results repositories
  - Distributed results visualization
- I want to focus where Fuego is different:
  - Repository of test definitions
  - Generalized output parsing system
  - Sharing of pass criteria and testcase documentation

Roadmap

Where is it going?

- Test definition improvements
- Modularization
- More tests
- Central server for sharing tests
- Interoperability with other systems

Next – test definitions

- Batch jobs for complex pipelines
  - Replacing current testplan system
- Migrate toward standard test definition format
  - *Requires defining a standard test definition*
- Support for provisioning

Next – more stuff

- Modularization
  - Separation of Fuego core from Jenkins
    - To use with other job schedulers, like LAVA
    - To use with other results visualizers, like Squad
  - Create a standalone parser
- More tests
  - Particularly for LTS kernels
    - (Dependent on provisioning work)
- Central server for sharing tests
  - The “test store”
  - The “test dispatcher”

Test server concept

- Goal is to increase testing by creating a (large) distributed test network
- Most systems are based on LAB testing
  - Labs have specialized hardware for automation
- Fuego is focused on “board at desk” model
  - Model should scale better due to lack of HW requirement
    - Should scale to # of contributors to Linux
    - ie. Anyone should be able to run a test
- There will always be more nodes NOT connected to automation hardware than connected

Test server concept (cont.)

- Requires appropriate disposition of results
  - To support “mining” of results requires standardized results format
    - Including standardized testcase name space
  - or realtime feedback to test requestor
- Create a virtual distributed lab
  - Developer can request a test on hardware not normally available to them
  - Long-term goal is 10,000 nodes
  - Automatic routing of test requests to appropriate nodes

Next - interoperability

- Interoperability with other systems
  - Lab APIs
    - PDU standard
  - External hardware API
    - Probably start with power measurement monitor API
  - Re-using tests from other systems
  - Exporting tests to other systems

Legacy Roadmap items

- Documentation
  - Conversion to reStructuredText
    - Convert documentation from wiki to ReadTheDocs
  - Tutorials

We're going to keep working on it!!

Resources

- Fuego web server:
  - http://fuegotest.org/
  - wiki: http://fuegotest.org/wiki
- Mailing list:
  - https://lists.linuxfoundation.org/mailman/listinfo/fuego
- Repositories:
  - https://bitbucket.org/fuegotest/fuego
  - https://bitbucket.org/fuegotest/fuego-core


Fuego

Bonus Material

Why is it called “Fuego”?

- Fuego = Tierra del Fuego
  - One place where penguins live
- Fuego = Fire
  - Often associated with trials and purifying
- It sounds neat

Old roadmap (Dec 2017)

- Listed here to show changes in emphasis
- To give us a Roadmap scorecard

Old Roadmap (cont.)

- Near future (cont.):
  - Testplan enhancements
    - Controlled test sequences
      - Similar to Jenkins pipelines
      - Processing multiple steps (provisioning, testing, notifications, report generation) in sequence
    - More fields for plan configuration
    - *[This turned into the new batch job system, and is still in development]*
  - Report generator and more charting control
    - Now that we have unified output, we can do queries, and different output formats
    - *[Not much progress here]*

Old Roadmap (cont. 2)

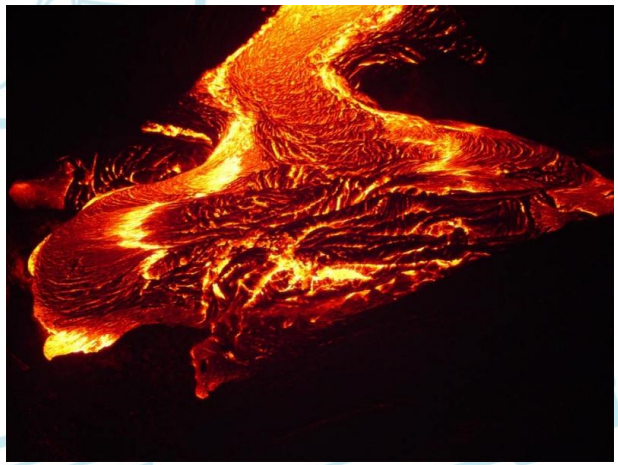
- Near future:
  - Documentation
    - Conversion to reStructuredText
    - Refactoring
    - Tutorials
  - New tests for AGL, LTSI, CIP
    - What tests to tackle next?

Old Roadmap (cont. 3)

- Near future (cont.):
  - System provisioning support
    - Install of software under test
      - Has been out-of-scope for Fuego
    - e.g. AGL image deploy, LTSI kernel update, etc.
    - Full automation requires board management API
    - Looking at labgrid as possible solution
  - *[No progress – blocked on new batch system]*
- Long-term
  - Distributed test network – [*some progress on fserver]*
  - Hardware testing – *[no progress]*

Other Priorities

- LAVA integration
  - We have everything needed for transport integration
  - Need test-level integration
    - Separate build phase
      - *[done]*
    - Deploy to LAVA server
      - *[prototyped?]*
    - Create LAVA test that does:
      - Execute test on board
      - Collect results
    - Execute Linaro test
      - *[prototyped?]*
  - *[some progress made]*



Previous roadmap scorecard

- Testplan enhancements — Almost done with phase 1
- Report generator — No progress
- Documentation improvements — No progress
- Distributed test network — Fserver code integrated
- Hardware testing — No progress- blocked on HW API
- LAVA Integration — Some prototypes written
- Release self-test — Stopped working – no maintenance

Obstacles to roadmap

- Fuego doesn't do provisioning
  - Automated installation

Dev board installation issues

- Many dev boards use SDcard design
- not set up for automated re-installation
- may not support top-of-tree (version gap)

Distributions don't support automated replacement of kernels

- Can't automate putting a top-of-tree kernel into an existing distribution
  - Due to:
    - Version gap
    - Distribution-specific patches
  - If you could automate this, you could automate distribution maintenance
- Current solution:
  - Put a new kernel on the board, and hope that nothing breaks
    - Hope that the parts that are incompatible are not under test
- Version gap is inevitable – that's a problem

Automated installation

- Dev board is not the final product
- Final product may not support software update
- Final product is locked down
  - Users not allowed to manipulate software
    - Can't install different kernel or test programs
- Final product may support OTA update
  - Requires setting up a service for OTA delivery
  - Limited permissions to install software
  - Requires special keys to install replacement software
- Automated re-installation of software is contrary to software security on the device
  - this is by design